Arduino Air Density Monitor with LCD Display – a.k.a. "Rho-Duino"
Nick Cinquino, 2/4/2015

The density of air, usually expressed in kilograms per cubic meter, symbolized by the Greek letter Rho, is a very important variable in the field of aerodynamics and appears in many equations. A few examples include force of drag, parachute descent rate, Pitot tube velocity measurements, and wing lift. Sometimes, people just throw in a value of 1.2 Kg/m3 without knowing the actual air density. That can obviously affect accuracy! If it's a hot/humid day that value isn't even close. In the following examples, Rho is the "squiggly P":

$$F_D = \tfrac{1}{2}\rho v^2 C_D A$$

$$v_e = \sqrt{\frac{2\,W_t}{S_o\,C_d\,\rho}}$$

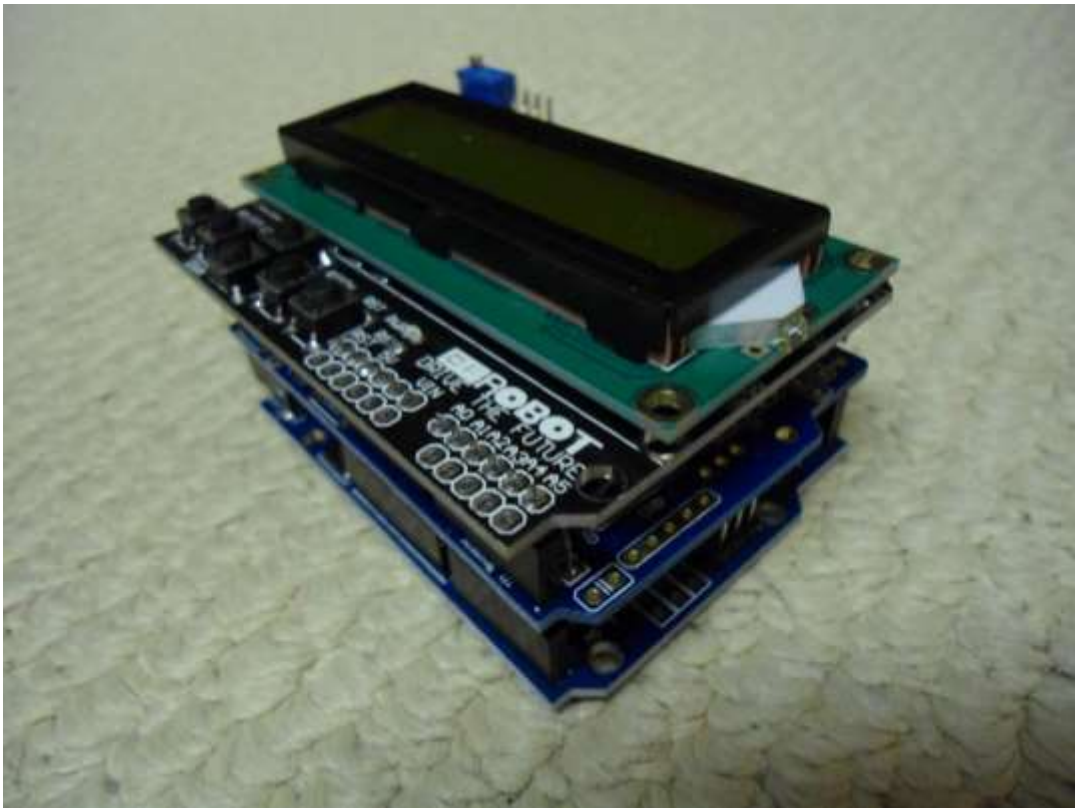
$$P_{total} - P_\infty = \frac{\rho}{2} V_\infty^2$$

$$V_\infty = \sqrt{\frac{2\times(P_{total} - P_\infty)}{\rho}}$$

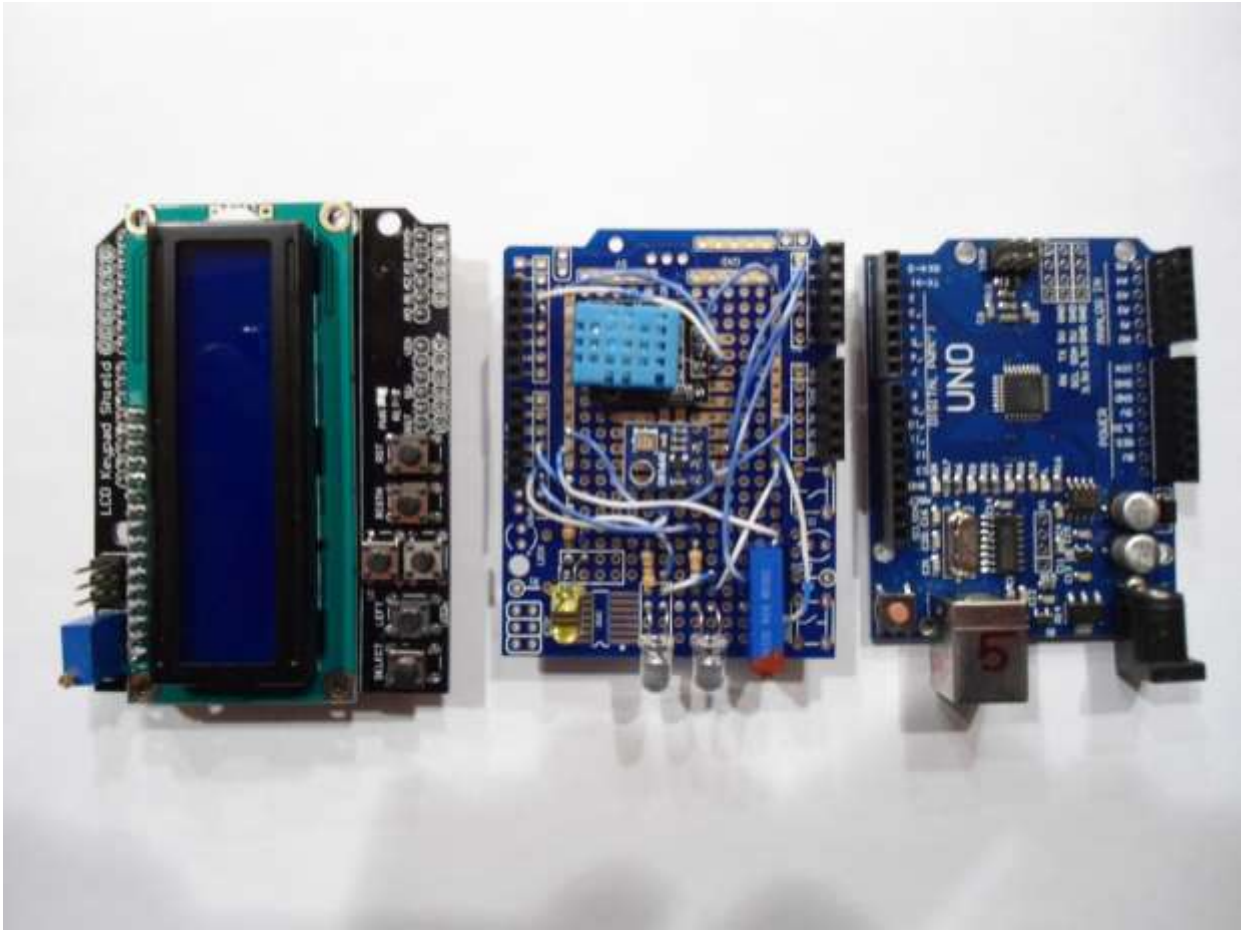
$$L_{ift} = C_L \times \tfrac{1}{2}\rho v^2 s$$

For Amateur Rocketry specifically, Rho is essential in the estimation of the altitude and velocity of a flight, and parachute descent rate.
The density of air can be calculated if one has data on air temperature, barometric "station" pressure, and humidity, preferably as dew point but relative humidity % can be used as well. Online calculators are available that return air density, but first the temp/pressure/RH data needs to be acquired, recorded and entered. Doing the calculations manually takes time and can be rage-inducing. It would be awfully convenient if Rho could be displayed in real time!

The Arduino microcontroller can do it all…take in the serial temperature, humidity and barometric pressure data, run all the calculations, including altitude and dew point, and display it all on a 16X2 LCD screen, with selectable pages. The Rho-Duino can (just barely) fit into a shirt pocket. If the shirt is a loud Hawaiian or bowling shirt, the bulge may be difficult to detect. Following are photos, circuit schematic and sketch for building a Rho-Duino.



The Rho-Duino, a triple-decker circuit with Arduino Uno at the bottom, custom circuit in the center and an LCD keypad shield on top. Lines are feedthroughs via the stacking headers.



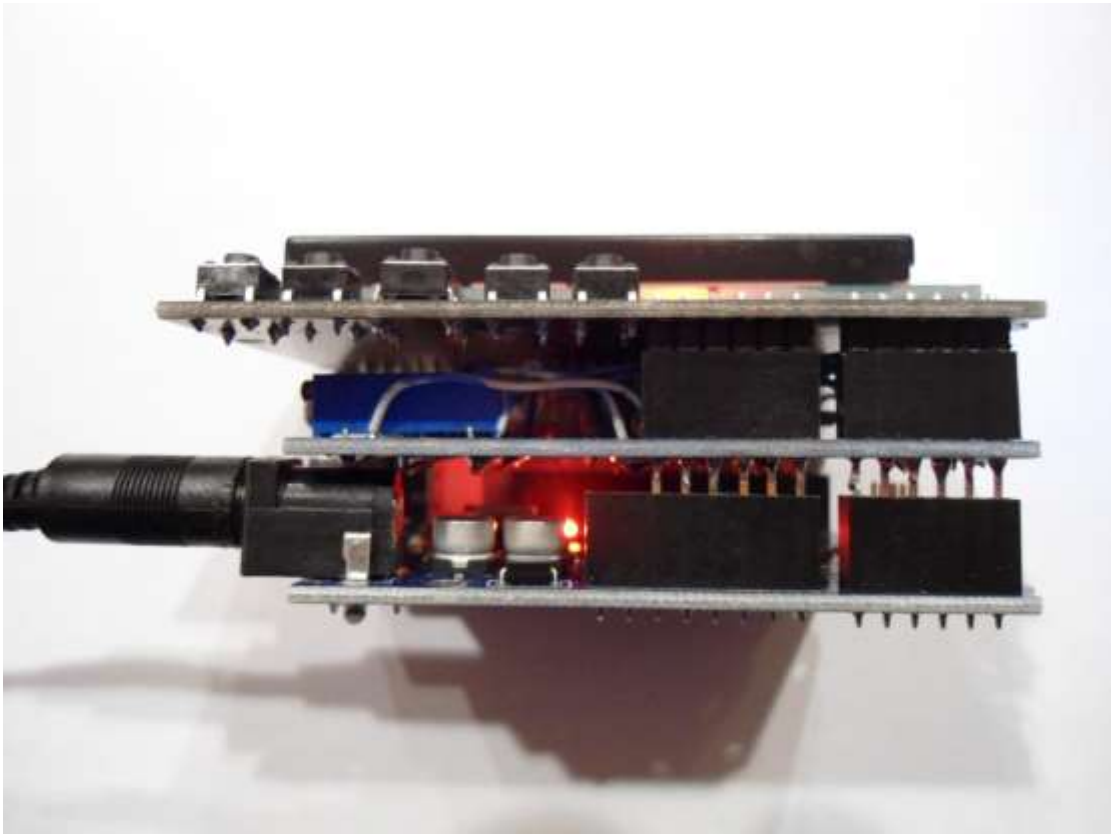
The 3 layers separated. In the center section, note the BMP180 pressure sensor , the DHT11 humidity sensor, and the 10-turn pot. Center section made on a “prototype shield board” with stacking headers. As the Arduino collects the temperature, humidity and pressure data, it uses the following equations to calculate air density in Kg/m3. The top equation with the fractional exponent is a real trip to work out manually. See references for additional info on air density calculation.

$$p_{\mathrm{sat}} = 6.1078 \times 10^{\frac{7.5T}{T+237.3}}$$

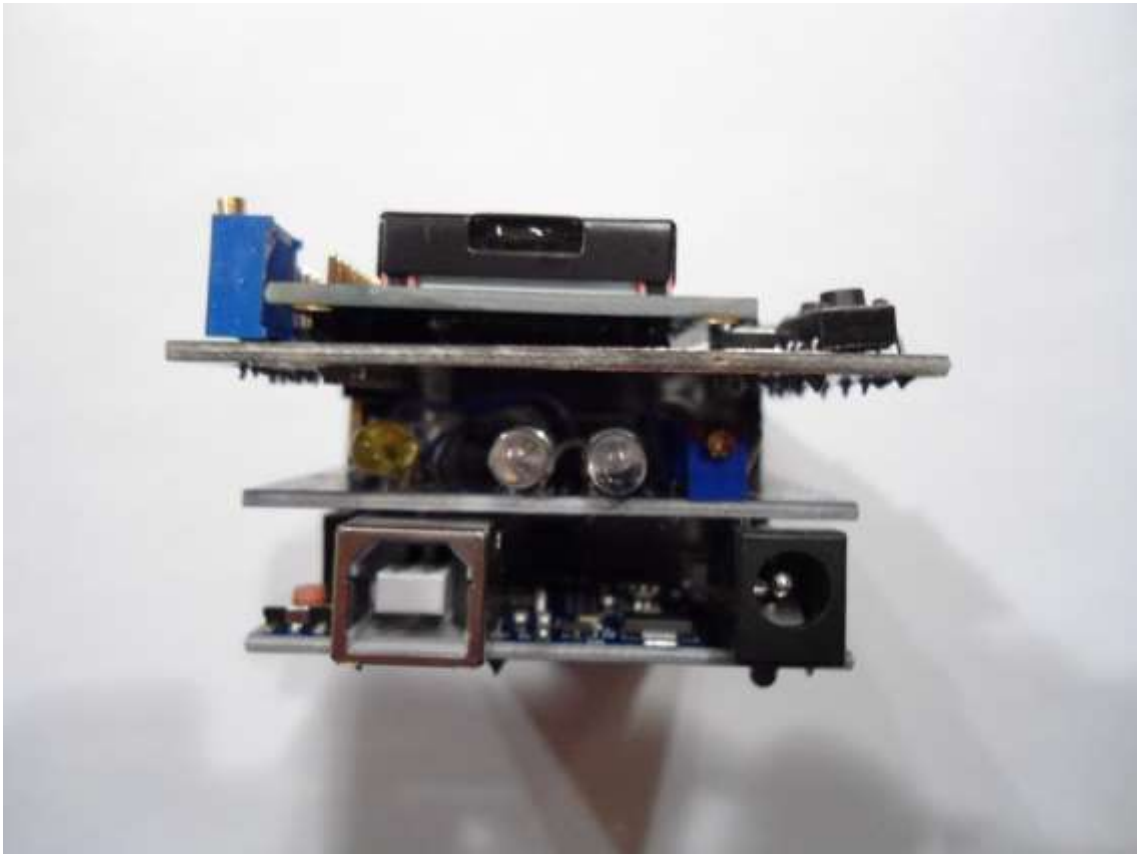
$$p_v = \phi p_{\mathrm{sat}}$$

$$p_d = p - p_v$$

$$\rho_{\mathrm{humid\ air}} = \frac{p_d}{R_d T} + \frac{p_v}{R_v T} = \frac{p_d M_d + p_v M_v}{RT}$$

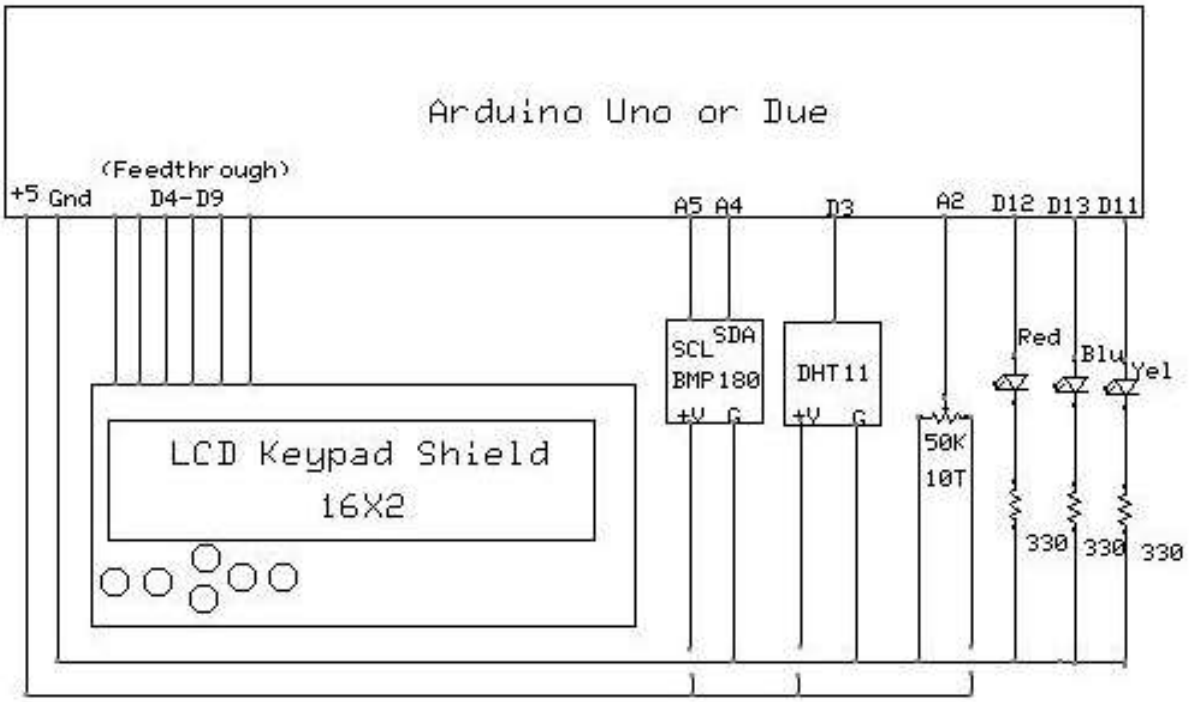


Side view of Rho-Duino. Note stacking headers. The blue pot is the Kollsman altitude adjustment. The keypad switches select different data display pages.



End view of Rho-Duino. Note multiturn potentiometer and LED's.

CIRCUIT SCHEMATIC:



ARDUINO SKETCH (paste into Arduino IDE window). This is a blend of 2 published sketches for the sensors, plus the author's for altitude and Rho calculations, display and keypad control:

```
//Rho-Duino NJC 1/29/15   -   DHT11 humidity and BMP180 baro/temp. Rho calculation.
//BMP180: Vcc to +5, Gnd to Gnd, SCL to A5, SDA to A4
//DHT11: + to +5, -to gnd, Data to Digital Pin3.


#include <Wire.h>
#include <DHT.h>
int analogKeypad=0;
float keycode=0;
#define DHTPIN 3  //sig into digitalpin 3, Vcc to +5 gnd to gnd
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
#define BMP085_ADDRESS 0x77  // I2C address of BMP085
const unsigned char OSS = 0;  // Oversampling Setting
#include <LiquidCrystal.h>
LiquidCrystal lcd(8,9,4,5,6,7);
float degf;
float kelvin=0;
float dewptk=0;
float dewptc=0;
float dewptf=0;
float baromb;          //pressure millibar
float baropascal;      //pressure in Pascals
float gconstdry = 287.058;      //gas constant for dry air
float gconstwaterv = 461.495;    //gas constant, water vapor
double psatmb;               //pressure saturated millibar
double psatnum;              //pressure saturated numerator
double psatden;              //pressure saturated denominator
double psatfract;            //pressure saturated fraction
double psatexpo;             //exponent for pressure saturated
double pvapwater;            //pressure, water vapor component
double pdryair;            //pressure, dry air
double psatpascals;          //pressure saturated in Pascals
double phumidA;              //part A of equation
double phumidB;              //part B of equation
double rho;                //variable for air density in Kg/m3
int analogPin=2;  //analog input at A2, use with LCD shield.
int val = 0;  //initialize value to 0
float pottopress;  //from (pot*0.0017578)+29.92
float pressalt;  //from (SLP - Baro) *994
int ledblu = 13;  //blue led, to flash at each update
```

```
// Calibration values, baro
int ac1;
int ac2;
int ac3;
unsigned int ac4;
unsigned int ac5;
unsigned int ac6;
int b1;
int b2;
int mb;
int mc;
int md;
// b5 is calculated in bmp085GetTemperature(...), this variable is also used in
bmp085GetPressure(...)
// so ...Temperature(...) must be called before ...Pressure(...).
long b5;
short temperature;
long pressure;



void setup()
{
  lcd.begin(16, 2);
  Serial.begin(9600);
  Wire.begin();
  bmp085Calibration();
  dht.begin();
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print(" Arduino/Keypad");
  lcd.setCursor(0,1);
  lcd.print(" WX Data NJC V8");
  delay(5000);
}



void loop()
{
  digitalWrite (ledblu,HIGH);  //flash the blue led to indicate update
  delay (50);
  digitalWrite (ledblu,LOW);
  val=analogRead(analogPin);
  pottopress=(val*0.0017578)+29.92;  //"Kollsman Window" value calculation
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  t=(t*1.8)+32;
```

```
if (isnan(t) || isnan(h))
{
  Serial.println("Failed to read from DHT");
}

delay(500);
float degc = temperature * 0.1;
degf = (degc*1.8) +32;
float inhg = pressure * 0.0003 *1.01465;
baromb = inhg *33.86;
baropascal = inhg * 3386;
kelvin=degc+273.15;
dewptk=kelvin-((100-h)/5);
dewptc=dewptk-273.15;
dewptf=((dewptc*1.8)+32);
temperature = bmp085GetTemperature(bmp085ReadUT());
pressure = bmp085GetPressure(bmp085ReadUP());
pressalt = (pottopress-inhg)*994;  //calculate feet
psatnum = 7.5 * degc;         //calculate the numerator of the psat exponent
psatden = degc+237.3;         //calculate the denominator of the psat exponent
psatfract = psatnum / psatden;  //calculate the psat exponent fraction
psatexpo = pow(10, psatfract);   //calculate exponent for psat
psatmb = 6.1087 * psatexpo;      //calculate psat in millibars
psatpascals = psatmb * 100;     //convert psat millibars (hectopascals) to pascals
pvapwater = (h/100*psatmb)*100;   //calculate water vapor pressure component
pdryair = baropascal - pvapwater;    //calculate partial pressure dry air component
phumidB = pdryair / (gconstdry * kelvin);  //part B of the pressure humid air calculation
phumidA = pvapwater / (gconstwaterv * kelvin);  //part A of humid air pressure
rho = phumidA + phumidB;  //calculate RHO in Kg/m3!
degf = (degc *1.8)+32;     //convert temp to F



analogKeypad=analogRead(0);
if ((analogKeypad>50)&&(analogKeypad<190))
{
  lcd.clear();
  lcd.setCursor(4,0);
  lcd.print("%RH  = ");
  lcd.print(h,0);
  lcd.setCursor(0,1);
  lcd.print("Dew Pt");
  lcd.print((char)223);
  lcd.print("F = ");
  lcd.print(dewptf,0);

}
```

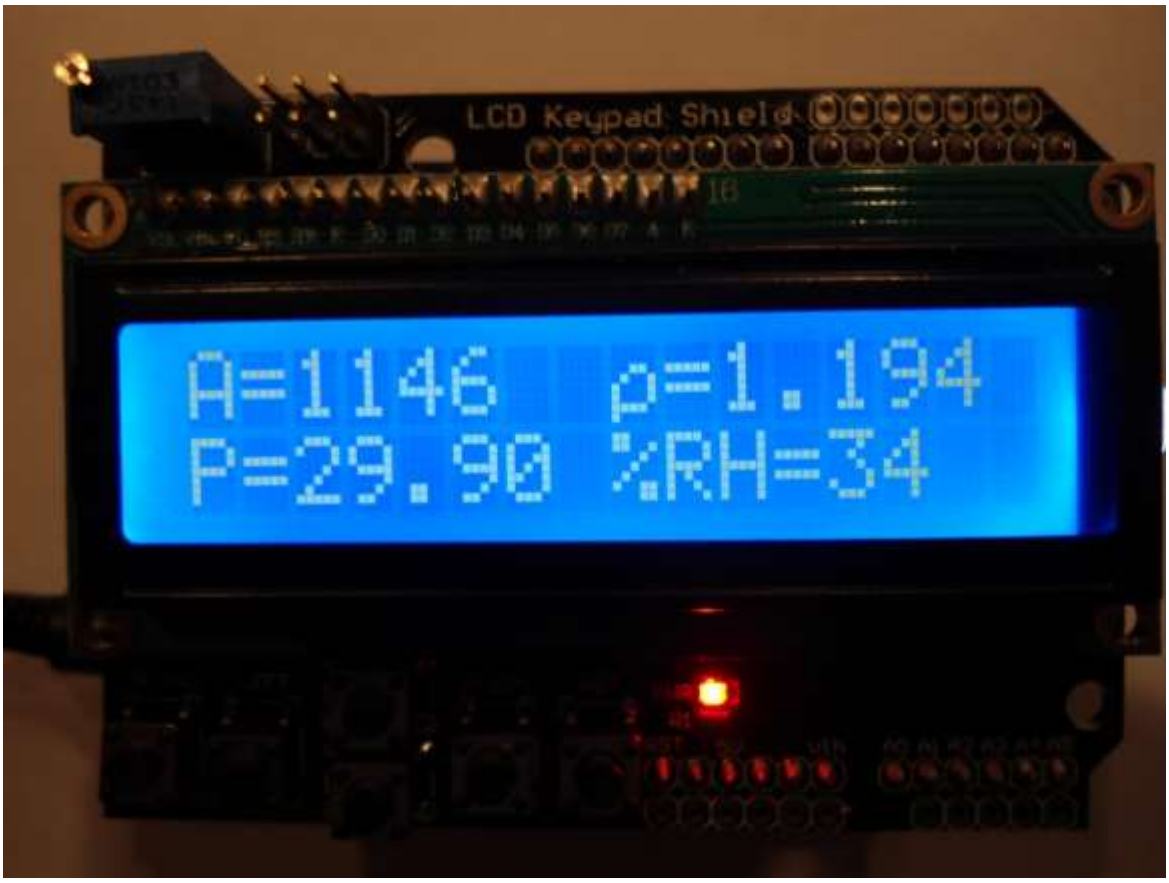
```
else if ((analogKeypad >=200)&&(analogKeypad <=380))
{
  lcd.clear();
  lcd.setCursor(3,0);
  lcd.print((char)223);
  lcd.print("F= ");
  lcd.print(degf,3);
  lcd.setCursor(0,1);
  lcd.print((char)223);
  lcd.print("C=");
  lcd.print(degc,1);
  lcd.setCursor(8,1);
  lcd.print((char)223);
  lcd.print("K=");
  lcd.print(kelvin,1);
}


else if ((analogKeypad >=390)&&(analogKeypad <=530))
{
  lcd.clear();
  lcd.setCursor(2,0);
  lcd.print("InHg= ");
  lcd.print(inhg,4);
  lcd.setCursor(2,1);
  lcd.print("  mB= ");
  lcd.print(baromb,2);
}


else if ((analogKeypad >=570)&&(analogKeypad <=780))
{
  lcd.clear();
  lcd.setCursor(1,0);
  lcd.print("Density, Kg/m3");
  lcd.setCursor(3,1);
  lcd.print((char)230);       //ASCII2 Character for Rho
  lcd.print("= ");
  lcd.print(rho,4);
}


else if (analogKeypad <=20)
{
  lcd.clear();
  lcd.setCursor(0,0);
```

```
  lcd.print("B=");
  lcd.print(inhg);
  lcd.setCursor(8,0);
  lcd.print("K=");
  lcd.print(pottopress,2);
  lcd.setCursor(3,1);
  lcd.print("Alt,'= ");
  lcd.print(pressalt,0);
}


else
{
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("A=");
  lcd.print(pressalt,0);
  lcd.setCursor(8,0);
  lcd.print((char)230);
  lcd.print("=");
  lcd.print(rho,3);
  lcd.setCursor(0,1);
  lcd.print("P=");
  lcd.print(inhg,2);
  lcd.setCursor(8,1);
  lcd.print("%RH=");
  lcd.print(h,0);
}



Serial.print(degf,2);
Serial.print(",   ");
Serial.print(inhg,3);
Serial.print(",   ");
Serial.print(h,1);
Serial.print(",   ");
Serial.print(t,1);
Serial.print(",   ");
Serial.print(dewptf,2);
Serial.print(",   ");
Serial.print(pottopress,2);
Serial.print(",   ");
Serial.print(pressalt,2);
Serial.print(",   ");
Serial.print(rho,4);
Serial.print(",");
Serial.println();
```

```
  delay(500);
}


// Stores all of the bmp085's calibration values into global variables
// Calibration values are required to calculate temp and pressure
// This function should be called at the beginning of the program
void bmp085Calibration()
{
  ac1 = bmp085ReadInt(0xAA);
  ac2 = bmp085ReadInt(0xAC);
  ac3 = bmp085ReadInt(0xAE);
  ac4 = bmp085ReadInt(0xB0);
  ac5 = bmp085ReadInt(0xB2);
  ac6 = bmp085ReadInt(0xB4);
  b1 = bmp085ReadInt(0xB6);
  b2 = bmp085ReadInt(0xB8);
  mb = bmp085ReadInt(0xBA);
  mc = bmp085ReadInt(0xBC);
  md = bmp085ReadInt(0xBE);
}
// Calculate temperature given ut.
// Value returned will be in units of 0.1 deg C
short bmp085GetTemperature(unsigned int ut)
{
  long x1, x2;
  x1 = (((long)ut - (long)ac6)*(long)ac5) >> 15;
  x2 = ((long)mc << 11)/(x1 + md);
  b5 = x1 + x2;
  return ((b5 + 8)>>4);
}
// Calculate pressure given up
// calibration values must be known
// b5 is also required so bmp085GetTemperature(...) must be called first.
// Value returned will be pressure in units of Pa.
long bmp085GetPressure(unsigned long up)
{
  long x1, x2, x3, b3, b6, p;
  unsigned long b4, b7;
  b6 = b5 - 4000;
  // Calculate B3
  x1 = (b2 * (b6 * b6)>>12)>>11;
  x2 = (ac2 * b6)>>11;
  x3 = x1 + x2;
  b3 = (((((long)ac1)*4 + x3)<<OSS) + 2)>>2;
  // Calculate B4
  x1 = (ac3 * b6)>>13;
  x2 = (b1 * ((b6 * b6)>>12))>>16;
```

```
  x3 = ((x1 + x2) + 2)>>2;
  b4 = (ac4 * (unsigned long)(x3 + 32768))>>15;
  b7 = ((unsigned long)(up - b3) * (50000>>OSS));
  if (b7 < 0x80000000)
    p = (b7<<1)/b4;
  else
    p = (b7/b4)<<1;
  x1 = (p>>8) * (p>>8);
  x1 = (x1 * 3038)>>16;
  x2 = (-7357 * p)>>16;
  p += (x1 + x2 + 3791)>>4;
  return p;
}
// Read 1 byte from the BMP085 at 'address'
char bmp085Read(unsigned char address)
{
  unsigned char data;
  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.write(address);
  Wire.endTransmission();
  Wire.requestFrom(BMP085_ADDRESS, 1);
  while(!Wire.available());
  return Wire.read();
}
// Read 2 bytes from the BMP085
// First byte will be from 'address'
// Second byte will be from 'address'+1
int bmp085ReadInt(unsigned char address)
{
  unsigned char msb, lsb;
  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.write(address);
  Wire.endTransmission();
  Wire.requestFrom(BMP085_ADDRESS, 2);
  while(Wire.available()<2);
  msb = Wire.read();
  lsb = Wire.read();
  return (int) msb<<8 | lsb;
}
// Read the uncompensated temperature value
unsigned int bmp085ReadUT()
{
  unsigned int ut;
  // Write 0x2E into Register 0xF4
  // This requests a temperature reading
  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.write(0xF4);
  Wire.write(0x2E);
```

```
  Wire.endTransmission();
  // Wait at least 4.5ms
  delay(5);
  // Read two bytes from registers 0xF6 and 0xF7
  ut = bmp085ReadInt(0xF6);
  return ut;
}
// Read the uncompensated pressure value
unsigned long bmp085ReadUP()
{
  unsigned char msb, lsb, xlsb;
  unsigned long up = 0;
  // Write 0x34+(OSS<<6) into register 0xF4
  // Request a pressure reading w/ oversampling setting
  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.write(0xF4);
  Wire.write(0x34 + (OSS<<6));
  Wire.endTransmission();
  // Wait for conversion, delay time dependent on OSS
  delay(2 + (3<<OSS));
  // Read register 0xF6 (MSB), 0xF7 (LSB), and 0xF8 (XLSB)
  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.write(0xF6);
  Wire.endTransmission();
  Wire.requestFrom(BMP085_ADDRESS, 3);
  // Wait for data to become available
  while(Wire.available() < 3);
  msb = Wire.read();
  lsb = Wire.read();
  xlsb = Wire.read();
  up = (((unsigned long) msb << 16) | ((unsigned long) lsb << 8) | (unsigned long) xlsb) >> (8-
OSS);
  return up;
}
```



Default page display. A=altitude, feet. Rho=air density, Kg/m3. P=baro InHg. %RH=humidity.

OPERATION:

Upon powerup, a 5-second title page is displayed on the LCD. This can be changed in the sketch. Next, the default page shows up, displaying the variables as shown in the default display pic above. When the LCD Keypad button on the far left is pressed and held down, a full display of rho only, 4-decimal places with unit is displayed. If the next button to the right is pressed and held down, the display shows barometric pressure in inches mercury and in millibars, the unit used on surface analysis weather charts. If the "up" button is pressed and held down, relative humidity % and calculated dew point is displayed. If the down button is pressed and held, the display shows temperature in degrees F, C and K. If the next button to the right is pressed and held, the display shows barometric pressure, InHg, the Kollsman setting, and altitude in feet. The keypad button on the far right is a reset; the Arduino resets and the display returns to the title page and on to the default page again.

For altimeter use, the Rho-Duino requires calibration with the Kollsman setting. Take the Rho-Duino to the nearest location of known elevation, in my case a local airport. Turn the Kollsman adjust pot until the altitude reading equals that known sea-level altitude. Immediately take the unit to another location, like home, and determine its elevation. Now that location is calibrated as well. From multiple tests I now believe my home is at 830' ASL. Without Rho-Duino I'd have never known that. It's best to do this on a day in which barometric changes are not changing rapidly. Keep in mind that barometric pressure is constantly changing; frequent altitude readjustment is necessary. When outdoors at a rocket flying field, keep Rho-Duino in the shade, away from direct sunlight. Allow 10 minutes for the humidity sensor to equilibrate, if moving from a cool, dry air conditioned vehicle (high Rho) to a hot humid flying field (low Rho).

NOTES:
The DHT.h library is the only additional library required. Download and add to your Arduino library folder, make sure it's all in a separate folder. Accuracy of the DHT11 is uncertain at very low %RH values. Comparing DHT11 readings to a pricey digital humidity meters shows a difference of a few %RH at low humidity, in the winter. The calculation of dewpoint from humidity and temperature is a shortcut method said to have good accuracy at dewpoints around 50F and above. For the calculation of Rho, the temperature, pressure and humidity are direct from the sensors, so the match to online Rho calculators, with the same input variables, is excellent. Note that digital pin 10 of the Arduino is unused. Keep it that way! Many of the low cost cloned LCD keypad shields use pin 10 as an LCD backlight brightness adjustment, but the circuit has a design flaw that could damage the Arduino. So, just don't do anything to pin 10 and that flaw is irrelevant. Altitude in feet is also simplified, based on 994' per InHg. This has good accuracy up to about 5000', beyond that the altitude reading will increasingly derange.

REFERENCES:

Richard Shelquist's Density Altitude site, excellent info, with calculators:
https://wahiduddin.net/calc/density_altitude.htm

Wikipedia's air density page:
http://en.wikipedia.org/wiki/Density_of_air

http://www.sanluisobispo.com/2012/09/01/2208988/air-density-is-a-heavy-matter.html

Another excellent air density calculator: http://www.baranidesign.com/air-density/air-density.htm

Rocket pitot tube article: **Sport Rocketry March / April 2014**
March / April Issue of Sport Rocketry Magazine. Editor`s Space, Measurement of the Indicated Airspeed of a Small Model Rocket with a Miniature Datalogger & Pitot Tube, Nick Cinquino and Leonard Johnson.